IN THE SPECIFICATION:

Please replace paragraph [0009] with the following amended paragraph:

[0009] One embodiment provides a standard application development template residing in a computer-readable medium and configured to facilitate application development. The standard application development template generally includes a plurality of front-end templates and at least one back-end template configured for specifying background processes of the application under development, the back-end template including at least one template agent. The plurality of front-end templates are configured for specifying user interface elements of an application under development, the ~~front-and~~ front-end generally include at least one of each of: a template form, a template view, and a template shared action.

Please replace paragraph [0025] with the following amended paragraph:

[0025] In one aspect, the invention is particularly advantageous in the context of highly modular languages such as object-oriented programming languages including ~~Java~~ the JAVA® and C++ programming languages. However, the use of object-oriented terminology is merely for convenience or illustration, and not limiting of the invention. Further, in some cases, different terms are used for the same or similar aspects of the invention. For example, in the claims appended below, the term “application”, "routine" or “function” may be used as a generic substitute for the term "method" to avoid any suggestion of that the invention is limited to particular object-oriented programming languages.

Please replace paragraph [0037] with the following amended paragraph:

[0037] In the depicted embodiment of FIG. 2, the front-end components of the standard development template 120 include a template form ~~211A~~ 211 ,template view ~~211B~~ 212 and template shared action ~~211C~~ 213 and the back-end components include a template agent ~~211D~~ 214. Forms (created using the template form ~~211A~~ 211) are design elements that accommodate the entering and retrieving of information from a

system such as Lotus Notes ®. Forms can be used to create documents and manage the data within documents. In some embodiments, forms can be used to create an interface through which database queries can be built and submitted. Views (created using the template view ~~211B~~ 212) are elements that provide a summarized view of data. For instance, in the Lotus Notes ® environment, views may provide a list of documents in a database that are usually sorted or categorized to make finding documents easier. Multiple views may be utilized to summarize and present the data residing in a database. Shared actions (created using the template shared action ~~211C~~ 213) are core elements of any object, such as a form. For example, when a user clicks a button on a form, a task is performed, such as closing the form or submitting data within the form to the database. Agents (created using the template agent ~~211D~~ 214) may be implemented as functions that execute an action or set of actions and may be configured to run as a background process. Agents may also be configured to interact with multiple applications. Illustrative actions performed by agents include: automatically removing out-dated documents in a database, or processing a group of documents in batch mode at the end of a business day.

Please replace paragraph [0048] with the following amended paragraph:

[0048] As described above, the components of the application may need to provide a wide variety of core functionality in addition to the project specific functionality. Accordingly, the front-end core script library 223 and back-end core script library 224 include a plurality of front-end components 309 and back-end components 310. For example, suppose one of the forms 302 in the user interface has a requirement to track when the document is saved. Rather than write custom code to track document save history, the developer can leverage the functionality provided by the Change Log class of the classes ~~309~~ 310 which is a component of the back-end core script library ~~310~~ 224. Similarly, any of the front-end components 309 or back-end components 310 belonging to the front-end core script library 223 and back-end core script library, respectively, can be leveraged during application development. In turn, these core libraries may leverage global script libraries 123 in order to reuse commonly needed

data structures and declarations. Table 1 provides a brief description of each of the classes belonging to the front-end core script library 223 and back-end core script library 224 in FIG. 3.

Please replace paragraph [0056] with the following amended paragraph:

[0056] At step 502 business requirements (or functional requirements) are gathered for the standard development template 120. Keeping in mind that the template is going to be used to create new applications that are configured for solving particular types of business problems, the basic requirements of the business problems are identified. For example, if the current standard development template 120 is going to be used to build content management based applications, some components the template might include are particular types of forms, views, shared actions, agents. One of the form templates, for example, may be designed to be used to build and submit queries related to content management.

Please replace paragraph [0063] with the following amended paragraph:

[0063] Recall that agents are considered back-end components and accordingly ~~the~~ following the creation of an agent from a template agent (640, 641, 642) it is determined at step 643 whether additional back-end functionality (not provided by the standard development template 120) is needed. If additional functionality is needed, at step 644 a check is performed to determine whether components providing the desired functionality exist in the back-end core script library 224 or global script library 123. If a suitable component is not found, new code is developed, tested and placed in the back-end project specific script library 222. In one embodiment, development of such new components is described with reference to FIG. 4. At step 646 the functionality is associated with the current back-end component. Upon completion of step 646 the next component is processed.